14 ~~17~~. A data processing system for executing a method having a plurality of paths, wherein the data processing system executes native machine code, the data processing system comprising:

Q3

identification means for identifying a path [within the method that is] being executed, <u>wherein the path is one of the plurality of paths in the method and</u> wherein a plurality of bytecodes are associated with the path; and

compilation means for compiling bytecodes for the path being executed, wherein the bytecodes are compiled into native machine code, wherein bytecodes for unexecuted paths <u>within the method</u> remain uncompiled.

REMARKS

Claims 1-32 are pending in the present application. Claims 1, 4, and 17 were amended. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 102, Anticipation

The examiner has rejected claims 1-32 under 35 U.S.C. § 102 as being anticipated by *Kolawa et al.* (U.S. Patent No. 5,784,553; hereinafter referred to as "*Kolawa*").This rejection is respectfully traversed.

In receiving the claims, the examiner stated:

> As per claim 1, Kolawa teaches **identifying a path within the method that is being executed** is shown in column 11 line 42-44 ("The dynamic symbolic execution is performing along the **path taken by an actual execution of the program"**), path taken by an actual execution of the program inherently including identifying a path within a method that is being executed, **plurality of instructions are associated with the paths** is shown in column 24 line 30-32 ("The TGS Driver Program executes the **program for the path** that corresponds to the input initially chosen and, **for all of the instructions found in the path"**), translating the first type instructions for the path being executed is shown in column 3 line 34-42 ("computer program written in the JAVA programming language, the **computer program being represented by JAVA bytecodes after being compiled by a JAVA compiler**. The method includes the steps of reading the JAVA bytecodes; obtaining an input value for the computer programs; **symbolically executing an instruction of the computer program represented in the JAVA bytecodes"**), compiling JAVA program into **Java bytecode**

> **inherently including instructions are translated into second type instruction as claimed, unexecuted path remain untranslated** is shown in column 5 line 53-55 ("The original source code 11 comprises all types of files **used to express an uncompiled, that is, non-object code**, computer program, including definitional and declarative files"), **uncompiled that is non-object code** inherently including unexecuted path remain untranslated as claimed.

Office Action dated March 2, 2000, pages 2-3

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983).

Each and every feature of the presently claimed invention is not found arranged as they are in the claims. Claim 1 reads as follows:

> 1. A process in a data processing system executing a routine having a plurality of paths, wherein the routine has includes a plurality of first type instructions and wherein the data processing system executes second type instructions, the process comprising:
>
> identifying a path within the routine that is being executed, wherein the path is one of the plurality of paths in the routine and wherein a plurality of first type instructions are associated with the path; and
>
> translating the first type instructions for the path being executed, wherein first type instructions are translated into second type instructions for execution by the data processing system, wherein first type instructions for unexecuted paths within the routine remain untranslated.

Claim 4 reads as follows:

> 4. A process in a data processing system for executing a method having a plurality of paths, wherein the data processing system executes native machine code, the process comprising:
>
> identifying a path within the method that is being executed, wherein the path is one of the plurality of paths in the within the method and wherein a plurality of bytecodes are associated with the path; and

A

compiling bytecodes for the path being executed, wherein the bytecodes are compiled into native machine code executed by the data processing system, wherein bytecodes for unexecuted paths within the method remain uncompiled.

Claim 24 reads as follows:

> 24. A data processing system comprising:
> a method having a plurality of paths in which each path within the plurality of paths contains a number of bytecodes;
> identification means for identifying that the method is to be executed; and
> compilation means for compiling the bytecodes into instructions for execution by the data processing system for each path within the plurality of paths as each path is executed.

The other independent claims include similar features. First, *Kolawa* does not teach the step of identifying a path being executed in which the path is part of a method or routine. The cited section of *Kolawa* reads as follows:

> The dynamic symbolic execution is performed along the path taken by an actual execution of the program under test which corresponds to some set of inputs **28** in the test suite database **12**.

Kolawa, column 11, lines 42-45.

This portion of *Kolawa* does not teach identifying a path in the routine being executed as in the presently claimed invention as believed by the examiner. This section merely recites that dynamic symbolic execution is performed along the path taken by an actual execution of the program under test. The examiner has misapplied the concept of "inherent" anticipation. Section 102 of Title 35 deals with novelty and loss of patent rights. An invention is said to be "anticipated" when it is squarely described or disclosed in a single reference as identified from one of the categories of 35 U.S.C. § 102, commonly referred to as "prior art". Express anticipation occurs when the invention is expressly disclosed in the prior art, patent or publication.

In some cases, however, when the claimed invention is not described *in haec verba*, the "doctrine of inherency" is relied on to establish anticipation. Under the principles of inherency, a claim is anticipated if a structure in the prior art necessarily functions in accordance with the limitations of a process or method claim. *In re King*,

A

801 F.2d 1324, 231 U.S.P.Q. 136 (Fed. Cir. 1986). A prior art reference that discloses all of a patent's claim limitations anticipates that claim even though the reference does not expressly disclose the "inventive concept" or desirable property the patentee discovered. *Verdgaal Brothers, Inc. v. Union Oil Company of California*, 814 F.2d 628, 2 U.S.P.Q.2d 1051, (Fed. Cir. 1987). It suffices that the prior art process inherently possessed at that property. *Id.* Mere possibilities or even probabilities, however, are not enough to establish inherency. The missing claimed characteristics must be a "natural result" flowing from what is disclosed. *Continental Can Co. v. Monsant Co.*, 948 F.2d 1264, 20 U.S.P.Q.2d 1746 (Fed. Cir. 1991). Unstated elements in a reference are inherent when they exist as a "matter of scientific fact". *Constant v. Advanced MicroDevices, Inc.*, 848 F.2d 1560, 7 U.S.P.Q.2d 1057 (Fed. Cir.), *cert. denied*, 488 U.S. 892 (1988) and *Hughes Aircraft Co. v. United States*, 8 U.S.P.Q.2d 1580 (Ct. Cl. 1988). Otherwise, the invention is not inherently anticipated.

In the present case, it does not necessarily follow that a path is identified within the routine being executed based on the cited section of *Kolawa*. This portion of *Kolawa* only recites that the dynamic execution is performed along a path that is taken by an actual execution. No identifying necessarily follows this statement. Next, the examiner cites the following for the translating step:

> Yet another embodiment of the present invention is a computer-implemented method of finding inputs that will generate runtime errors in a computer program written in the JAVA programming language, the computer program being represented by JAVA bytecodes after being compiled by a JAVA compiler. The method includes the steps of reading the JAVA bytecodes; obtaining an input value for the computer program; symbolically executing an instruction of the computer program represented in bytecodes;

Kolawa, column 3, lines 34-42. The examiner cites the following for the proposition that *Kolawa* teaches the feature of unexecuted paths remaining untranslated:

> The original source code **11** comprises all types of files used to express an uncompiled, that is, non-object code, computer program, including definitional and declarative files.

Kolawa, column 5, lines 53-55. In the present case, the examiner's assertions that the elements of the present invention are present can be made only through the use of the

A

applicants' disclosure as a template to fill in the missing elements. In actuality, the examiner has selected portions of *Kolawa* without regard to their actual arrangement. Even if, *arguendo*, the cited portions of *Kolawa* teach the steps asserted by the examiner, these steps are not disclosed in the same arrangement as in the presently claimed invention in independent claims 1 and 4. This situation is apparent when *Kolawa* is viewed in its entirety rather than in the piece meal fashion as presented by the examiner. For example, the identifying step, translating step, and the feature of unexecuted paths remaining untranslated are based on piece meal selections from *Kolawa*.

Kolawa states:

> Yet another embodiment of the present invention is a computer-implemented method of finding inputs that will generate runtime errors in a computer program written in the JAVA programming language, the computer program being represented by JAVA bytecodes after being compiled by a JAVA compiler. The method includes the steps of reading the JAVA bytecodes; obtaining an input value for the computer program; symbolically executing an instruction of the computer program represented in bytecodes; examining all of the symbolic expressions on which the instruction depends based on the symbolic execution; and storing the input value in a test database and marking the input value as an input that generates a runtime error when a solution to the symbolic expressions generates an error condition.

Kolawa, column 3, lines 33-47. A review of *Kolawa* for what it actually teaches is that the program is compiled to create JAVA bytecodes, based on this teaching of *Kolawa*, portions of the program would not remain untranslated because *Kolawa* expressly teaches in the section cited by the examiner that the program is compiled by a JAVA compiler. Nowhere does the cited reference teach or does it necessarily follow that portions of the program remain uncompiled. The examiner has only pointed to a section that teaches compiling the program and a section that states that the original source code is an uncompiled non-object code computer program. It is consistent in *Kolawa* that the original program is uncompiled and that compiling the program generates JAVA bytecodes. However, it does not necessarily follow that only portions of the program remain uncompiled. In fact, the teaching of compiling the program teaches that no uncompiled bytecodes remain.

A

The portion of *Kolawa* cited by the examiner for identifying a path describes dynamic symbolic execution of the program. In the presently claimed invention, this step takes place prior to the translating. According to *Kolawa*, the dynamic symbolic execution occurs after the entire program is compiled or translated. Thus, in contrast to identifying then translating, this portion of *Kolawa* teaches that the symbolic execution occurs after the program has been compiled – translated from one type of instruction to another type of instruction.

Further, this teaching in *Kolawa* does not provide any teaching or suggestion of translating the instructions from one type to another for a path being executed. The presently claimed invention translates instructions from one type into another as being executed. Based on the teaching of *Kolawa*, the translation of instructions occur through a compiling process, which does not involve execution of the program. Thus, when *Kolawa* is viewed as a whole for what it teaches, rather than in a piece meal fashion, each and every feature of the presently claimed invention is not shown in *Kolawa*, as arranged in claims 1 and 4.

Further, the other independent claims and the other dependent claims are patentable over this cited reference for the same reasons. Further, the dependent claims include other additional combinations of features not taught or suggested by the reference. For example, claims 3, 6, and 13 include an additional step of storing the compiled instructions in an execution order. The examiner has made the same inherency argument as before. It does not necessarily follow that translated instructions will be stored in an order of execution. Depending on the implementation, the instructions may be stored in the same order in which they are found in the method or routine. Consequently, it is respectfully urged that the rejection of claims 1-32 under 35 U.S.C. § 102 have been overcome.

Further, the presently claimed invention in claims 1-32 are not obvious in view of *Kolawa*. No teaching, suggestion, or incentive is present for one of ordinary skill in the art to make the necessary changes to *Kolawa* to reach the presently claimed invention. When *Kolawa* is viewed as a whole, *Kolawa* teaches compiling or translating a program in its entirety. The symbolic execution, the portion on which the examiner relies on for

A

the alleged identifying of the path, occurs after the compilation. Therefore, *Kolawa* teaches away from the presently claimed invention if it were assumed that the symbolic execution as cited by the examiner included identifying a path that is being executed. As discussed above, such a teaching or interpretation is not present in *Kolawa* expressly or inherently.

Further, one of ordinary skill in the art would not be motivated to make these changes when the problems addressed by *Kolawa* and the present invention are considered by one of ordinary skill in the art. The present invention recognizes that problems exist with just in time compiling in which entire methods are compiled regardless of whether they are used. *See* Specification, page 5, lines 1-3. The presently claimed invention is concerned with the problem of trying to reduce use of data processing system resources during execution of methods. In contrast, *Kolawa* is concerned with difficulties in testing software and in particular creating a good test suite. *Kolawa*, column 1, lines 28-46. Thus, one of ordinary skill in the art would not be motivated to look *Kolawa*, much less modify *Kolawa* with respect to the presently claimed invention.

Therefore, the presently claimed invention may be reached only through an improper use of hindsight with the benefit of applicant's invention as a template to piece together and interpret teachings from *Kolawa*. No teaching, suggestion, or incentive is present for the interpretations or modifications needed to reach the presently claimed invention.

A

II. **Conclusion**

It is respectfully urged that the subject application is patentable over *Kolawa* and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: July 11, 2000

Respectfully submitted,

[illegible]

Duke W. Yee
Reg. No. 34,285
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicant

A